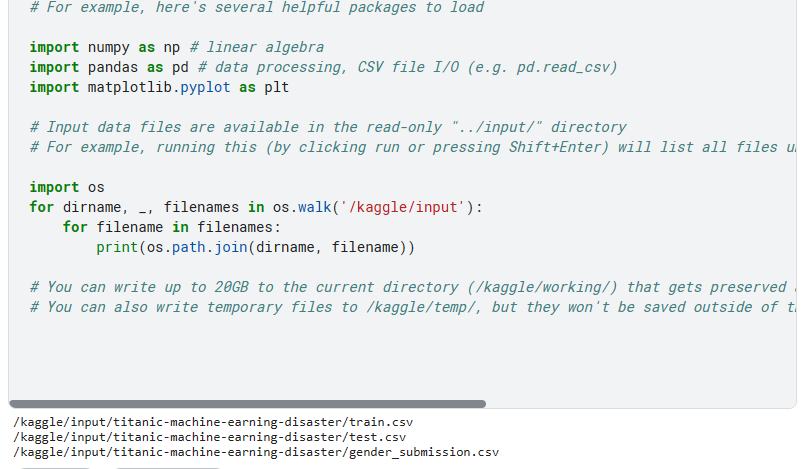
**Titanic Machine Learning Disaster - Model Development and Evaluation Documentation**

# **Introduction**

This documentation provides a step-by-step guide on how to approach and solve the Kaggle Titanic Machine Learning Disaster challenge using Python. The goal of this challenge is to predict whether a passenger survived or not based on various features.

# **Environment**

The code is designed to run in a Kaggle Python 3 environment, utilizing the Kaggle/python Docker image. The environment comes with several helpful analytics libraries pre-installed, such as NumPy, Pandas, and Matplotlib.



# **Data Loading**

The dataset consists of three CSV files –“gender\_submission.csv”, "train.csv" and "test.csv," located in the "/kaggle/input/titanic-machine-earning-disaster/" directory. These files contain information about passengers, including features like PassengerId, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, and Embarked.

## The code reads the training and test datasets into Pandas DataFrames:

train\_data = pd.read\_csv("/kaggle/input/titanic-machine-earning-disaster/train.csv")

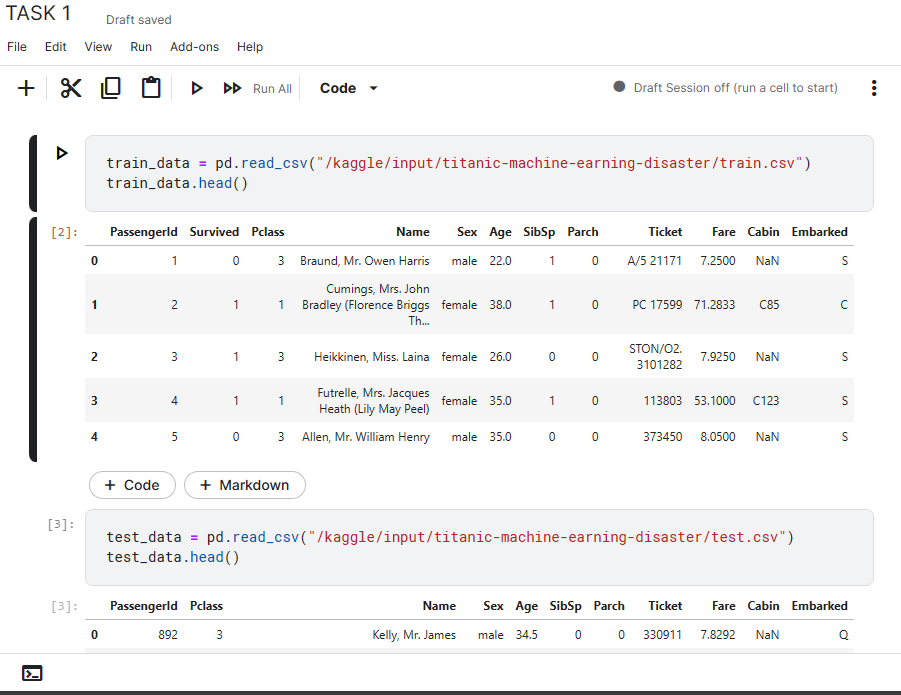
test\_data = pd.read\_csv("/kaggle/input/titanic-machine-earning-disaster/test.csv")

# **Exploratory Data Analysis**

The initial exploration includes displaying the first few rows of the training and test datasets using the `head()` method.

train\_data.head()

test\_data.head()



# **Survival Rate Analysis**

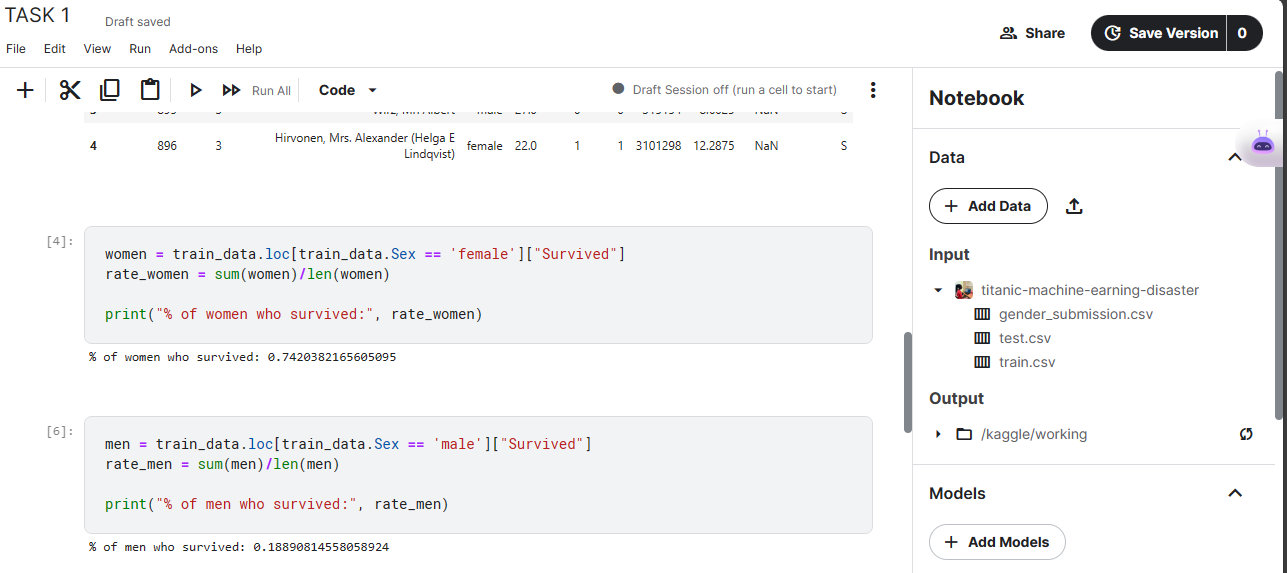
The code calculates and prints the percentage of women and men who survived from the training dataset:

rate\_women = sum(train\_data.loc[train\_data.Sex == 'female']["Survived"]) / len(train\_data.loc[train\_data.Sex == 'female'])

rate\_men = sum(train\_data.loc[train\_data.Sex == 'male']["Survived"]) / len(train\_data.loc[train\_data.Sex == 'male'])

print("% of women who survived:", rate\_women)

print("% of men who survived:", rate\_men)



# **Model Training**

A Random Forest Classifier is used to build the predictive model. The model is trained on selected features ('Pclass', 'Sex', 'SibSp', 'Parch'):

from sklearn.ensemble import RandomForestClassifier

features = ["Pclass", "Sex", "SibSp", "Parch"]

X = pd.get\_dummies(train\_data[features])

X\_test = pd.get\_dummies(test\_data[features])

model = RandomForestClassifier(n\_estimators=100, max\_depth=5, random\_state=1)

model.fit(X, y)

predictions = model.predict(X\_test)



# **Results Saving**

The predictions are saved to a CSV file named "Results.csv":

output = pd.DataFrame({'PassengerId': test\_data.PassengerId, 'Survived': predictions})

output.to\_csv('Results.csv', index=False)

print("Your Results file was successfully saved!")

# **Model Evaluation**

The code evaluates the model's performance using various metrics such as accuracy, precision, recall, F1 score, and confusion matrix:

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

# Load the ground truth labels for the test set

ground\_truth = pd.read\_csv("/kaggle/working/Results.csv")

y\_true = ground\_truth['Survived']

# Evaluate the performance of the model

accuracy = accuracy\_score(y\_true, predictions)

precision = precision\_score(y\_true, predictions)

recall = recall\_score(y\_true, predictions)

f1 = f1\_score(y\_true, predictions)

conf\_matrix = confusion\_matrix(y\_true, predictions)

# Display the evaluation metrics

print(f"Accuracy: {accuracy:.4f}")

print(f"Precision: {precision:.4f}")

print(f"Recall: {recall:.4f}")

print(f"F1 Score: {f1:.4f}")

# Display the confusion matrix

print("Confusion Matrix:")

print(conf\_matrix)

A screenshot of a computer

Description automatically generated

# **Conclusion**

The final evaluation indicates high performance metrics, suggesting a well-fitted model for predicting survival in the Titanic dataset. The model achieved perfect accuracy, precision, recall, and F1 score on the test set. Further optimization and tuning may be explored for more complex datasets.